#### Draft Text Specification

#### 7.3.2.2 Sequence parameter set RBSP syntax

|  |  |
| --- | --- |
| seq\_parameter\_set\_rbsp( ) { | Descriptor |
| … |  |
| **intra\_block\_copy\_enabled\_flag** | u(1) |
| if( intra\_block\_copy\_enabled\_flag ) |  |
| **sps\_nonsquare\_intra\_bc\_enabled\_flag** | u(1) |
| **implicit\_rdpcm\_enabled\_flag** | u(1) |
| … |  |

**sps\_extension\_flag**[ 0 ] equal to 1 specifies that transform\_skip\_rotation\_enabled\_flag, transform\_skip\_context\_enabled\_flag, intra\_block\_copy\_enabled\_flag, sps\_nonsquare\_intra\_bc\_enabled\_flag, implicit\_rdpcm\_enabled\_flag, explicit\_rdpcm\_enabled\_flag, extended\_precision\_processing\_flag, intra\_smoothing\_disabled\_flag, high\_precision\_offsets\_enabled\_flag and fast\_rice\_adaptation\_enabled\_flag are present in the SPS RBSP syntax structure. sps\_extension\_flag[ 0 ] equal to 0 specifies that these syntax elements are not present.

**sps\_nonsquare\_intra\_bc\_enabled\_flag** equal to 1 specifies that intra lock copy for partition of size PART\_2NxN or PART\_Nx2N may be invoked in the decoding process for intra prediction. sps\_nonsquare\_intra\_bc\_enabled\_flag equal to 0 specifies that intra block copy for partitions of size PART\_2NxN and PART\_Nx2N is not applied. When not present, the value of sps\_nonsquare\_intra\_bc\_enabled\_flag is inferred to be equal to 0.

**7.3.8.5 Coding unit syntax**

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize ) { | Descriptor |
| … |  |
| if( intra\_block\_copy\_enabled\_flag ) |  |
| **intra\_bc\_flag**[ x0 ][ y0 ] | ae(v) |
| if( !intra\_bc\_flag[ x0 ][ y0 ] && slice\_type != I ) ~~{~~ |  |
| ~~if( slice\_type != I )~~ |  |
| **pred\_mode\_flag** | ae(v) |
| if( (sps\_nonsquare\_intra\_bc\_enabled\_flag && intra\_bc\_flag[ x0][ y0] ) | | CuPredMode[ x0 ][ y0 ] != MODE\_INTRA | | log2CbSize = = MinCbLog2SizeY ) |  |
| **part\_mode** | ae(v) |
| ~~}~~ |  |
| if( CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA ) { |  |
| if( PartMode = = PART\_2Nx2N && pcm\_enabled\_flag &&   !intra\_bc\_flag[ x0 ][ y0 ] &&   log2CbSize >= Log2MinIpcmCbSizeY &&  log2CbSize <= Log2MaxIpcmCbSizeY ) |  |
| **pcm\_flag**[ x0 ][ y0 ] | ae(v) |
| if( pcm\_flag[ x0 ][ y0 ] ) { |  |
| while( !byte\_aligned( ) ) |  |
| **pcm\_alignment\_zero\_bit** | f(1) |
| pcm\_sample( x0, y0, log2CbSize ) |  |
| } else if( intra\_bc\_flag[ x0 ][ y0 ] ) { |  |
| mvd\_coding( x0, y0, 2) |  |
| if( sps\_nonsquare\_intra\_bc\_enabled\_flag ) { |  |
| if( PartMode == Part\_2NxN ) { |  |
| mvd\_coding( x0, y0 + ( nCbS/2 ), 2 ) |  |
| } else if( PartMode == Part\_Nx2N ){ |  |
| mvd\_coding( x0 + ( nCbS/2 ), y0, 2 ) |  |
| } |  |
| } |  |
| } else { |  |
| … |  |

**part\_mode** specifies partitioning mode of the current coding unit. The semantics of part\_mode depend on CuPredMode[ x0 ][ y0 ]. The variables PartMode and IntraSplitFlag are derived from the value of part\_mode as defined in Table 7‑10.

The value of part\_mode is restricted as follows:

* If CuPredMode[ x0 ][ y0 ] is equal to MODE\_INTRA, the following applies: ~~part\_mode shall be equal to 0 or 1.~~
* If intra\_bc\_flag[x0][y0] is equal to 0, part\_mode shall be equal to 0 or 1.
* Otherwise (intra\_bc\_flag[x0][y0] is equal to 1), part\_mode shall be in the range of 0 to 2, inclusive.
* Otherwise (CuPredMode[ x0 ][ y0 ] is equal to MODE\_INTER), the following applies:
* If log2CbSize is greater than MinCbLog2SizeY and amp\_enabled\_flag is equal to 1, part\_mode shall be in the range of 0 to 2, inclusive, or in the range of 4 to 7, inclusive.
* Otherwise, if log2CbSize is greater than MinCbLog2SizeY and amp\_enabled\_flag is equal to 0, or log2CbSize is equal to 3, part\_mode shall be in the range of 0 to 2, inclusive.
* Otherwise (log2CbSize is greater than 3 and less than or equal to MinCbLog2SizeY), the value of part\_mode shall be in the range of 0 to 3, inclusive.

When part\_mode is not present, the variables PartMode and IntraSplitFlag are derived as follows:

* PartMode is set equal to PART\_2Nx2N.
* IntraSplitFlag is set equal to 0.

Table 7‑10 – Name association to prediction mode and partitioning type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CuPredMode[ x0 ][ y0 ]** | **intra\_bc\_flag[x0][y0]** | **part\_mode** | **IntraSplitFlag** | **PartMode** |
| MODE\_INTRA | 0 | 0 | 0 | PART\_2Nx2N |
| 1 | 1 | PART\_NxN |
| 1 | 0 | 0 | PART\_2Nx2N |
| 1 | 1 | PART\_2NxN |
| 2 | 1 | PART\_Nx2N |
| MODE\_INTER | - | 0 | 0 | PART\_2Nx2N |
| 1 | 0 | PART\_2NxN |
| 2 | 0 | PART\_Nx2N |
| 3 | 0 | PART\_NxN |
| 4 | 0 | PART\_2NxnU |
| 5 | 0 | PART\_2NxnD |
| 6 | 0 | PART\_nLx2N |
| 7 | 0 | PART\_nRx2N |

#### 9.3.3.5 Binarization process for part\_mode

Inputs to this process are a request for a binarization for the syntax element part\_mode a luma location ( xCb, yCb ), specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture, and a variable log2CbSize specifying the current luma coding block size.

Output of this process is the binarization of the syntax element.

The binarization for the syntax element part\_mode is specified in Table 9‑36 depending on the values of CuPredMode[ xCb ][ yCb ] and log2CbSize.

**Table 9‑36 – Binarization for part\_mode**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CuPredMode**[ xCb ][ yCb ] | **intra\_bc\_flag [xCb][yCb]** | **part\_mode** | **PartMode** | **Bin string** | | | |
| log2CbSize >  MinCbLog2SizeY | | log2CbSize  = =  MinCbLog2SizeY | |
| !amp\_enabled\_flag | amp\_enabled\_flag | log2CbSize  = =  3 | log2CbSize > 3 |
| MODE\_INTRA | 0 | 0 | PART\_2Nx2N | - | - | 1 | 1 |
| 1 | PART\_NxN | - | - | 0 | 0 |
| 1 | 0 | PART\_2Nx2N | 1 | 1 | 1 | 1 |
| 1 | PART\_2NxN | 01 | 01 | 01 | 01 |
| 2 | PART\_Nx2N | 00 | 00 | 00 | 00 |
| MODE\_INTER | - | 0 | PART\_2Nx2N | 1 | 1 | 1 | 1 |
| 1 | PART\_2NxN | 01 | 011 | 01 | 01 |
| 2 | PART\_Nx2N | 00 | 001 | 00 | 001 |
| 3 | PART\_NxN | - | - | - | 000 |
| 4 | PART\_2NxnU | - | 0100 | - | - |
| 5 | PART\_2NxnD | - | 0101 | - | - |
| 6 | PART\_nLx2N | - | 0000 | - | - |
| 7 | PART\_nRx2N | - | 0001 | - | - |

### General decoding process for coding units coded in intra prediction mode

Inputs to this process are:

– a luma location ( xCb, yCb ) specifying the top-left sample of the current luma coding block relative to the top‑left luma sample of the current picture,

– a variable log2CbSize specifying the size of the current luma coding block.

Output of this process is a modified reconstructed picture before deblocking filtering.

The derivation process for quantization parameters as specified in subclause 8.6.1 is invoked with the luma location ( xCb, yCb ) as input.

A variable nCbS is set equal to 1  <<  log2CbSize.

Depending on the values of pcm\_flag[ xCb ][ yCb ] and IntraSplitFlag, the decoding process for luma samples is specified as follows:

– If pcm\_flag[ xCb ][ yCb ] is equal to 1, the reconstructed picture is modified as follows:

SL[ xCb + i ][ yCb + j ] =   
 pcm\_sample\_luma[ ( nCbS \* j ) + i ]  <<  ( BitDepthY − PcmBitDepthY ), with i, j = 0..nCbS − 1 (8‑12)

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0), if IntraSplitFlag is equal to 0, the following ordered steps apply:

1. When intra\_bc\_flag[ xCb ][ yCb ] is equal to 0, the derivation process for the intra prediction mode as specified in subclause 8.4.2 is invoked with the luma location ( xCb, yCb ) as input.
2. When intra\_bc\_flag[ xCb ][ yCb ] is equal to 1, the derivation process for block vector components in intra block copying prediction mode as specified in subclause 8.4.4 is invoked with the luma location ( xCb, yCb ) and variable log2CbSize as inputs, and the output being ~~bvIntra~~bvIntraArray.
3. The general decoding process for intra blocks as specified in subclause 8.4.4.1 is invoked with the luma location ( xCb, yCb ), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeY[ xCb ][ yCb ], the variable predModeIntraBc set equal to intra\_bc\_flag[ xCb ][ yCb ], the variable bvIntra set equal to bvIntraArray[ 0 ], the variable bvIntraArray, and the variable cIdx set equal to 0 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0 and IntraSplitFlag is equal to 1), for the variable blkIdx proceeding over the values 0..3, the following ordered steps apply:

1. When intra\_bc\_flag[ xCb ][ yCb ] is equal to 1, bvIntra is derived as follows:
   * If PartMode is equal to PART\_2NxN and blkIdx is equal to 2 or 3, bvIntra is set equal to bvIntraArray[1]
   * Otherwise, if PartMode is equal to PART\_Nx2N and blkIdx is equal to 1 or 3, bvIntra is set equal to bvIntraArray[1]
   * Otherwise, bvIntra is set equal to bvIntraArray[0]
2. The variable xPb is set equal to xCb + ( nCbS  >>  1 ) \* ( blkIdx % 2 ).
3. The variable yPb is set equal to yCb + ( nCbS  >>  1 ) \* ( blkIdx / 2 ).
4. The derivation process for the intra prediction mode as specified in subclause 8.4.2 is invoked with the luma location ( xPb, yPb ) as input.
5. The general decoding process for intra blocks as specified in subclause 8.4.4.1 is invoked with the luma location ( xPb, yPb ), the variable log2TrafoSize set equal to log2CbSize − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeY[ xPb ][ yPb ], the variable predModeIntraBc set equal to ~~0~~ intra\_bc\_flag[ xCb ][ yCb ], the variable bvIntra, the variable bvIntraArray, and the variable cIdx set equal to 0 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

When ChromaArrayType is not equal to 0, the following applies.

The variable log2CbSizeC is set equal to log2CbSize − ( ChromaArrayType  = =  3 ? 0 : 1 ).

Depending on the value of pcm\_flag[ xCb ][ yCb ] and IntraSplitFlag, the decoding process for chroma samples is specified as follows:

– If pcm\_flag[ xCb ][ yCb ] is equal to 1, the reconstructed picture is modified as follows:

SCb[ xCb / SubWidthC + i ][ yCb / SubHeightC + j ] =  pcm\_sample\_chroma[ ( nCbS / SubWidthC \* j ) + i ]  <<  
 ( BitDepthC − PcmBitDepthC ), with i = 0..nCbS / SubWidthC − 1, and j = 0..nS / SubHeightC − 1 (8‑13)

SCr[ xCb / SubWidthC + i ][ yCb / SubHeightC + j ] = pcm\_sample\_chroma[ ( nCbS / SubWidthC \* ( j + nCbS / SubHeightC ) ) + i ]  <<  
 ( BitDepthC − PcmBitDepthC ), with i = 0..nCbS / SubWidthC − 1, and j = 0..nS / SubHeightC − 1 (8‑14)

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0), if IntraSplitFlag is equal to 0 or ChromaArrayType is not equal to 3, the following ordered steps apply:

1. When intra\_bc\_flag[ xCb ][ yCb ] is equal to 0, the derivation process for the chroma intra prediction mode as specified in 8.4.3 is invoked with the luma location ( xCb, yCb ) as input, and the output is the variable IntraPredModeC.
2. The general decoding process for intra blocks as specified in subclause 8.4.4.1 is invoked with the chroma location ( xCb / SubWidthC, yCb / SubHeightC ), the variable log2TrafoSize set equal to log2CbSizeC, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeC, the variable predModeIntraBc set equal to intra\_bc\_flag[ xCb ][ yCb ], the variable bvIntra set equal to bvIntraArray[ 0 ], the variable bvIntraArray, and the variable cIdx set equal to 1 as inputs, and the output is a modified reconstructed picture before deblocking filtering.
3. The general decoding process for intra blocks as specified in subclause 8.4.4.1 is invoked with the chroma location ( xCb / SubWidthC, yCb / SubHeightC ), the variable log2TrafoSize set equal to log2CbSizeC, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeC, the variable predModeIntraBc set equal to intra\_bc\_flag[ xCb ][ yCb ], the variable bvIntra set equal to bvIntraArray[ 0 ], the variable bvIntraArray, and the variable cIdx set equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0, IntraSplitFlag is equal to 1 and ChromaArrayType is equal to 3), for the variable blkIdx proceeding over the values 0..3, the following ordered steps apply:

1. When intra\_bc\_flag[ xCb ][ yCb ] is equal to 1, bvIntra is derived as follows:
   * If PartMode is equal to PART\_2NxN and blkIdx is equal to 2 or 3, bvIntra is set equal to bvIntraArray[1]
   * Otherwise, if PartMode is equal to PART\_Nx2N and blkIdx is equal to 1 or 3, bvIntra is set equal to bvIntraArray[1]
   * Otherwise, bvIntra is set equal to bvIntraArray[0]
2. The variable xBS is set equal to xC + ( nS  >>  1 ) \* ( blkIdx % 2 ).
3. The variable yBS is set equal to yC + ( nS  >>  1 ) \* ( blkIdx / 2 ).
4. The derivation process for the chroma intra prediction mode as specified in 8.4.3 is invoked with the luma location ( xBS, yBS ) as input, and the output is the variable IntraPredModeC.
5. The general decoding process for intra blocks as specified in subclause 8.4.4.1 is invoked with the chroma location ( xBS, yBS), the variable log2TrafoSize set equal to log2CbSizeC − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeC, the variable predModeIntraBc set equal to ~~0~~ intra\_bc\_flag[ xCb ][ yCb ], the variable bvIntra, the variable bvIntraArray, and the variable cIdx set equal to 1 as inputs, and the output is a modified reconstructed picture before deblocking filtering.
6. The general decoding process for intra blocks as specified in subclause 8.4.4.1 is invoked with the chroma location ( xBS, yBS), the variable log2TrafoSize set equal to log2CbSizeC − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeC, the variable predModeIntraBc set equal to ~~0~~ intra\_bc\_flag[ xCb ][ yCb ], the variable bvIntra, the variable bvIntraArray, and the variable cIdx set equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

### Derivation process for block vector components in intra block copying prediction mode

Inputs to this process are:

* a luma location ( xCb, yCb ) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
* a variable log2CbSize specifying the size of the current luma coding block.

Output of this process is the block vector ~~bvIntra~~array bvIntraArray.

The variable nCbS is set equal to 1  <<  log2CbSize.

The variable BvpIntra[ compIdx ] specifies a block vector predictor. The horizontal block vector component is assigned compIdx = 0 and the vertical block vector component is assigned compIdx = 1.

Depending on the PartMode, the variable numPUs is derived as follows:

* numPUs = ( PartMode == PART\_2Nx2N ? 1 : 2 )

For the variable tempIdx proceeding over the value 0..numPUs-1, the following ordered steps apply:

* Depending on the value of PartMode, the variable xPb, yPb, xPbs, and yPbs are derived as follows:
  + If PartMode is equal to PART\_2Nx2N:

xPb = xCb

yPb = yCb

xPbs = nCbs

yPbs = nCbs

* + Otherwise, if PartMode is equal to PART\_Nx2N or PART\_2NxN:

xPb = xCb + ( nCbs >> 1 ) \* ( PartMode == PART\_2NxN ? 0 : tempIdx )

yPb = yCb+ ( nCbs >> 1 ) \* ( PartMode == PART\_2NxN ? tempIdx : 0 )

xPbs = nCbs >> ( PartMode == PART\_2NxN ? 0 : 1 )

yPbs = nCbs >> ( PartMode == PART\_2NxN ? 1 : 0 )

* The block vector ~~bvIntra~~array bvIntraArray is derived by the following ordered steps, for the variable compIdx proceeding over the values 0..1:

1. Depending upon the number of times this process has been invoked for the current coding tree unit, the following applies:

* If this process is invoked for the first time for the current coding tree unit, ~~bvIntra[ compIdx ]~~ bvIntraArray[ tempIdx ][ compIdx ] is derived as follows:

~~bvIntra[ compIdx ] = BvdIntra[ xCb ][ yCb ][ compIdx ] (8‑25)~~

bvIntraArray[ tempIdx ][ compIdx ] = BvdIntra[ xPb ][ yPb ][ compIdx ] (8‑25)

* Otherwise, ~~bvIntra[ compIdx ]~~bvIntraArray[ tempIdx ][ compIdx ] is derived as follows:

~~bvIntra[ compIdx ] = BvdIntra[ xCb ][ yCb ][ compIdx ] + BvpIntra[ compIdx ] (8‑25)~~

bvIntraArray[ tempIdx ][ compIdx ] = BvdIntra[ xPb ][ yPb ][ compIdx ] + BvpIntra[ compIdx ] (8‑25)

1. The value of BvpIntra[ compIdx ] is updated to be equal to ~~bvIntra[ compIdx ]~~ bvIntraArray[ tempIdx ][ compIdx ].

* It is a requirement of bitstream conformance that all of the the following conditions are true:

– The value of ~~bvIntra[ 0 ]~~bvIntraArray[ tempIdx ][ 0 ] shall be greater than or equal to – ( ~~xCb~~xPb % CtbSizeY + CtbSizeY ).

– The value of ~~bvIntra[ 1 ]~~bvIntraArray[ tempIdx ][ 1 ] shall be greater than or equal to – ( ~~yCb~~yPb % CtbSizeY ).

– When the derivation process for z-scan order block availability as specified in subclause 6.4.1 is invoked with ( xCurr, yCurr ) set equal to ~~( xCb, yCb )~~( xPb + ( nCbs >> 1 ) \* ( PartMode == PART\_2NxN ? 1 : 0 ), yPb + ( nCbs >> 1 ) \* ( PartMode == PART\_Nx2N ? 1 : 0) ) and the neighbouring luma location ( xNbY, yNbY ) set equal to ~~( xCb + bvIntra[ 0 ], yCb + bvIntra[ 1 ] )~~ ( xPb + bvIntraArray[ tempIdx ][ 0 ],  yPb + bvIntraArray[ tempIdx ][ 1 ] ) as inputs, the output shall be equal to TRUE.

– When the derivation process for z-scan order block availability as specified in subclause 6.4.1 is invoked with ( xCurr, yCurr ) set equal to ~~( xCb, yCb )~~( xPb + ( nCbs >> 1 ) \* ( PartMode == PART\_2NxN ? 1 : 0 ), yPb + ( nCbs >> 1 ) \* ( PartMode == PART\_Nx2N ? 1 : 0) ) and the neighbouring luma location ( xNbY, yNbY ) set equal to ~~( xCb + bvIntra[ 0 ] + nCbS − 1, yCb + bvIntra[ 1 ] + nCbS – 1 )~~( xPb + bvIntraArray[ tempIdx ][ 0 ] + xPbs − 1, yPb + bvIntraArray[ tempIdx ][ 1 ] + yPbs – 1 ) as inputs, the output shall be equal to TRUE.

– One or both of the following conditions shall be true:

~~– bvIntra[ 0 ] + nCbS <= 0~~

* bvIntraArray[ tempIdx ][ 0 ] + xPbs >> ( PartMode == PART\_2NxN ? 1 : 0 ) <= 0

~~– bvIntra[ 1 ] + nCbS <= 0~~

* bvIntraArray[ tempIdx ][ 1 ] + yPbs >> ( PartMode == PART\_Nx2N ? 1 : 0 ) <= 0

#### General decoding process for intra blocks

Inputs to this process are:

– a sample location ( xTb0, yTb0 ) specifying the top-left sample of the current transform block relative to the top‑left sample of the current picture,

– a variable log2TrafoSize specifying the size of the current transform block,

– a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding unit,

– a variable predModeIntra specifying the intra prediction mode,

– a variable predModeIntraBc specifying the intra block copying mode,

– a variable bvIntra specifying the intra block copying vector,

* a variable bvIntraArray specifying the array of intra block copying vectors within current CU,

– a variable cIdx specifying the colour component of the current block.

Output of this process is a modified reconstructed picture before deblocking filtering.

The luma sample location ( xTbY, yTbY ) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture is derived as follows:

( xTbY, yTbY ) = ( cIdx  = =  0 ) ? ( xTb0, yTb0 ) : ( xTb0 \* SubWidthC, yTb0 \* SubHeightC ) (8‑26)

The variable splitFlag is derived as follows:

– If cIdx is equal to 0, splitFlag is set equal to split\_transform\_flag[ xTbY ][ yTbY ][ trafoDepth ].

– Otherwise, if all of the following conditions are true, splitFlag is set equal to 1.

* cIdx is greater than 0
* split\_transform\_flag[ xTbY ][ yTbY ][ trafoDepth ] is equal to 1
* log2TrafoSize is greater than 2

– Otherwise, splitFlag is set equal to 0.

When predModeIntraBc is equal to 1, ChromaArrayType is not equal to 3, PartMode is not equal to PART\_2Nx2N and cIdx is larger than 0, the following applies:

* If trafoDepth is equal to 0, the variable xTbTem is set equal to xTb0, and yTbTem is set equal to yTb0.
* Otherwise, If trafoDepth is equal to 1, bvIntra is derived as follows:
  + If PartMode is equal to PART\_2NxN and yTb0 is not equal to yTbTem or if PartMode is equal to PART\_Nx2N and xTb0 is not equal to xTbTem, the following applies:

bvIntra = bvIntraArray[ 1 ]

Depending on the value of splitFlag, the following applies:

– If splitFlag is equal to 1, the following ordered steps apply:

1. The variables xTb1 and yTb1 are derived as follows:
   * The variable xTb1 is set equal to xTb0 + ( 1  <<  ( log2TrafoSize − 1 ) ).
   * The variable yTb1 is set equal to yTb0 + ( 1  <<  ( log2TrafoSize − 1 ) ).
2. The general decoding process for intra blocks as specified in this subclause is invoked with the location ( xTb0, yTb0 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.
3. The general decoding process for intra blocks as specified in this subclause is invoked with the location ( xTb1, yTb0 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.
4. The general decoding process for intra blocks as specified in this subclause is invoked with the location ( xTb0, yTb1 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.
5. The general decoding process for intra blocks as specified in this subclause is invoked with the location ( xTb1, yTb1 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.

– Otherwise (splitFlag is equal to 0), for the variable blkIdx proceeding over the values 0..( cIdx > 0  &&  ChromaArrayType  = =  2 ? 1 : 0 ), the following ordered steps apply:

1. The variable nTbS is set equal to 1  <<  log2TrafoSize.
2. The variable yTbOffset is set equal to blkIdx \* nTbS.
3. The variable yTbOffsetY is set equal to yTbOffset \* SubHeightC.
4. The variable residualDpcm is derived as follows:
   * If all of the following conditions are true, residualDpcm is set equal to 1.
     + implicit\_rdpcm\_enabled\_flag is equal to 1.
     + either transform\_skip\_flag[ xTbY ][ yTbY + yTbOffsetY ][ cIdx ] is equal to 1, or cu\_transquant\_bypass\_flag is equal to 1.
     + either predModeIntra is equal to 10, or predModeIntra is equal to 26.
   * Otherwise, residualDpcm is set equal to explicit\_rdpcm\_flag[ xTbY ][ yTbY + yTbOffsetY ][ cIdx ].
5. Depending upon the value of predModeIntraBc, the following applies:

– When predModeIntraBc is equal to 0, the general intra sample prediction process as specified in subclause 8.4.4.2.1 is invoked with the transform block location ( xTb0, yTb0 + yTbOffset ), the intra prediction mode predModeIntra, the transform block size nTbS, and the variable cIdx as inputs, and the output is an (nTbS)x(nTbS) array predSamples.

– Otherwise (predModeIntraBc is equal to 1), the intra block copying process as specified in subclause 8.4.4.2.7 is invoked with the transform block location ( xTb0, yTb0 + yTbOffset ), the transform block size nTbS, the variable bvIntra, the variable bvIntraArray, the variable trafoDepth, and the variable cIdx as inputs, and the output is an (nTbS)x(nTbS) array predSamples.

1. The scaling and transformation process as specified in subclause 8.6.2 is invoked with the luma location ( xTbY, yTbY + yTbOffsetY ), the variable trafoDepth, the variable cIdx, and the transform size trafoSize set equal to nTbS as inputs, and the output is an (nTbS)x(nTbS) array resSamples.
2. When residualDpcm is equal to 1, depending upon the value of predModeIntraBc, the following applies:

– When predModeIntraBc is equal to 0, the directional residual modification process for blocks using a transform bypass as specified in subclause 8.6.5 is invoked with the variable mDir set equal to predModeIntra / 26, the variable nTbS, and the (nTbS)x(nTbS) array r set equal to the array resSamples as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.

– Otherwise, (predModeIntraBc is equal to 1), the directional residual modification process for blocks using a transform bypass as specified in subclause 8.6.5 is invoked with the variable mDir set equal to explicit\_rdpcm\_dir\_flag[ xTbY ][ yTbY + yTbOffsetY ][ cIdx ], the variable nTbS, and the (nTbS)x(nTbS) array r set equal to the array resSamples as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.

1. When luma\_chroma\_prediction\_enabled\_flag is equal to 1, ChromaArrayType is equal to 3, and cIdx is not equal to 0, the residual modification process for transform blocks using luma-chroma prediction as specified in subclause 8.6.6 is invoked with the current luma transform block location ( xTbY, yTbY ), the variable nTbS, the variable cIdx, the (nTbS)x(nTbS) array rY set equal to the corresponding luma residual sample array resSamples of the current transform block, and the (nTbS)x(nTbS) array r set equal to the array resSamples as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.
2. The picture reconstruction process prior to in-loop filtering for a colour component as specified in subclause 8.6.6 is invoked with the transform block location ( xTb0, yTb0 + yTbOffset ), the variables nCurrSw and nCurrSh both set equal to nTbS, the variable cIdx, the (nTbS)x(nTbS) array predSamples, and the (nTbS)x(nTbS) array resSamples as inputs.

##### Specification of intra block copying prediction mode

Inputs to this process are:

– a sample location ( xTbCmp, yTbCmp ) specifying the top-left sample of the current transform block relative to the top‑left sample of the current picture,

– a variable nTbS specifying the transform block size,

– a variable bvIntra specifying the block copying vector,

* a variable bvIntraArray specifying the array of intra block copying vectors within current CU,
* a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding unit,

– a variable cIdx specifying the colour component of the current block.

Output of this process is the predicted samples predSamples[ x ][ y ], with x, y = 0..nTbS − 1.

If PartMode is not equal to PART\_2Nx2N and trafoDepth is equal to 0, for the variable tempIdx proceeding over the values 0..1, the following applies:

* + xTbCmp = xTbCmp + ( nTbs >> 1 ) \* ( PartMode == PART\_2NxN ? 0 : tempIdx )
  + yTbCmp = yTbCmp + ( nTbs >> 1 ) \* ( PartMode == PART\_2NxN ? tempIdx : 0 )
* The general intra block copying process as specified in this subclause is invoked with the location ( xTbCmp , yTbCmp ), the variable nTbS set equal to nTbS >> 1, the variable bvIntra is set equal to bvIntraArray[ tempIdx ], the variable bvIntraArray, the variable trafoDepth is set equal to 1, and the variable cIdx as inputs, and the output is an (nTbS)x(nTbS) array predSamples.

Otherwise, the following ordered steps apply:

1. The variable bv representing the block vector for prediction in full-sample units is derived as follows:

bv[ 0 ] = bvIntra[ 0 ] >> ( ( ( cIdx = = 0 ) ? 1 : SubWidthC ) − 1 ) (8‑63)

bv[ 1 ] = bvIntra[ 1 ] >> ( ( ( cIdx = = 0 ) ? 1 : SubHeightC ) − 1 ) (8‑64)

1. The (nTbS)x(nTbS) array of predicted samples samples, with x, y = 0..nTbS − 1, is derived as follows:

– The reference sample location (xRefCmp, yRefCmp ) is specified by:

( xRefCmp, yRefCmp ) = ( xTbCmp + x + bv[ 0 ], yTbCmp + y + bv[ 1 ] ) (8‑65)

* Each sample at the location ( xRefCmp, yRefCmp ) is assigned to predSamples[ x ][ y ].